

170929 逆向-Reversing.kr (Ransomware)

原创

奈沙夜影 于 2017-09-29 20:28:06 发布 604 收藏

分类专栏: [CrackMe](#)

版权声明: 本文为博主原创文章, 遵循 [CC 4.0 BY-SA](#) 版权协议, 转载请附上原文出处链接和本声明。

本文链接: <https://blog.csdn.net/whklhjh/article/details/78137138>

版权



[CrackMe 专栏收录该内容](#)

83 篇文章 2 订阅

订阅专栏

1625-5 王子昂 总结《2017年9月29日》【连续第362天总结】

A. Reversing.kr-Ransomware

B.

Ransomware

readme提示解密文件, 运行一堆乱码, 估计可能是韩文吧

查壳发现有UPX

用ESP定律手脱下来以后拖入IDA发现main函数块巨大, 反编译会一直等待

拖入OD进行跟踪, 发现在exit前, 位于44ab75的call会飞

跟进去以后是大片的花指令:

```
LCG - 主线程, 模块 - run
004135E0  55          push ebp
004135E1  8BEC       mov ebp, esp
004135E3  83EC 24    sub esp, 0x24
004135E6  53         push ebx
004135E7  56         push esi
004135E8  57         push edi
004135E9  60         pushad
004135EA  61         popad
004135EB  90         nop
004135EC  50         push eax
004135ED  58         pop eax
004135EE  53         push ebx
004135EF  5B         pop ebx
004135F0  60         pushad
004135F1  61         popad
004135F2  90         nop
004135F3  50         push eax
004135F4  58         pop eax
004135F5  53         push ebx
004135F6  5B         pop ebx
004135F7  60         pushad
004135F8  61         popad
004135F9  90         nop
004135FA  50         push eax
004135FB  58         pop eax
```

源程序只有10K, OD拖下来就高达10MB了, 本来以为UPX压缩效率惊人到这个地步, 原来都是这花指令再捣鬼

(PS: 用UPXShell拖下来只有300K, 代码倒是没少啥, 不过似乎有一些对空余地方的去除, 希望有懂的师傅指导一下QAQ)

估计这就是妨碍IDA反编译的东西了

先往下跟，发现在0044A775的地方开始正常代码

printf和scanf的前后大量调用了sub_401000，而这个函数跟进去也同是大片的花指令

长度高达0x10000左右，为了防止中间加载有效操作，将反汇编代码复制出来进行清洗，发现啥都没剩\(_ _)就是个空函数

大概流程是得到输入，然后open提供的file文件进行操作

试着重新建立了一个file文件，在其中写入00000等试验代码，随便输入一个key，待程序运行完成后再打开发现已经变成乱码了.....

居然是直接对源文件操作的，那我之前乱输的key不就直接让file文件乱掉了吗！

不备份真是恶劣.....还好rar中可以再找回来

那么为了使IDA正常工作，我们把这两段花指令NOP掉

从下到上全部NOP，然后在401000处写入C3 (retn) 使sub_401000能正常返回

保存以后拖入IDA会报库函数未识别的错，把相应的地方修改HEX糊弄过去先，得到整体框架再说：

我这边反编译出的代码缺陷很大，不过依靠文件名字符串识别出scanf和输入字符串的位置，就可以找到下边的关键代码了：

```
39 for ( *(_DWORD *) (a1 - 8) = 0; *(_DWORD *) (a1 - 8) < *(_DWORD *) (a1 - 16); ++*(_DWORD *) (a1 - 8) )
40 {
41     byte_541588[*(_DWORD *) (a1 - 8)] ^= input[*(_DWORD *) (a1 - 8) % *(_DWORD *) (a1 - 12)];
42     sub_401000();
43     byte_541588[*(_DWORD *) (a1 - 8)] = ~byte_541588[*(_DWORD *) (a1 - 8)];
44     sub_401000();
45 }
```

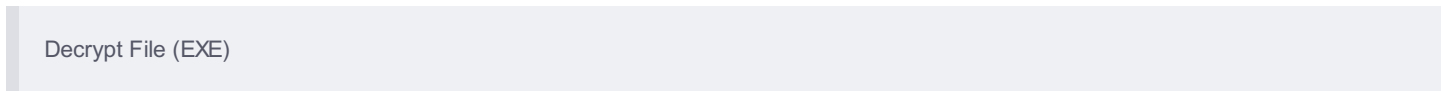
<http://blog.csdn.net/whklhxxx>

可以看出来，解密操作是将input逐个与二进制流异或，然后取反

不过没有提供密钥，所以有点懵逼

参考了一下别人的WriteUp，发现这个加密文件其实是个PE格式的EXE

提示在readme里：



我靠我以为EXE的意思是用EXE解密呢.....原来是告诉我们File本来是个EXE啊(\` \ `)' _ _ _

既然这样的话，通过PE文件的特定格式就能找到一些端倪了

比方说magicnumber，不过它有些短，所以当密钥太长的话就无能为力了

PE头中有一个dos stub程序，一般情况下运行会显示”this program cannot be run in the dos mode”这个字符串就足够长，可以用来破译了

还可以使用在程序末尾出现的00，它们足够长，而且源码相同易于识别：

```
000022c0h: E0 F0 FB F5 EF 97 9A 8C E9 EB F3 FF 8C 8F 93 9E ; 囧 飾殞殒?審撻
000022d0h: 86 9C 97 9A 8C 8C 93 9A 8B 8C 8F 93 9E 86 9C 97 ; 吮櫛瘡摻媽瓠瀾漣
000022e0h: 9A 8C 8C 93 9A 8B 8C 8F 93 9E 86 9C 97 9A 8C 8C ; 殞憲臻審撻吮櫛瘡
000022f0h: 93 9A 8B 8C 8F 93 9E 86 9C 97 9A 8C 8C 93 9A 8B ; 摻媽瓠瀾漣殞憲臻
00002300h: 8C 8F 93 9E 86 9C 97 9A 8C 8C 93 9A 8B 8C 8F 93 ; 審撻吮櫛瘡摻媽瓠
00002310h: 9E 86 9C 97 9A 8C 8C 93 9A 8B 8C 8F 93 9E 86 9C ; 瀾漣殞憲臻審撻吮
00002320h: 97 9A 8C 8C 93 9A 8B 8C 8F 93 9E 86 9C 97 9A 8C ; 櫛瘡摻媽瓠瀾漣殞
00002330h: 8C 93 9A 8B 8C 8F 93 9E 86 9C 97 9A 8C 8C 93 9A ; 憲臻審撻吮櫛瘡摻
00002340h: 8B 8C 8F 93 9E 86 9C 97 9A 8C 8C 93 9A 8B 8C 8F ; 媽瓠瀾漣殞憲臻審
00002350h: 93 9E 86 9C 97 9A 8C 8C 93 9A 8B 8C 8F 93 9E 86 ; 撻吮櫛瘡摻媽瓠瀾
00002360h: 9C 97 9A 8C 8C 93 9A 8B 8C 8F 93 9E 86 9C 97 9A ; 漣殞憲臻審撻吮櫛
00002370h: 8C 8C 93 9A 8B 8C 8F 93 9E 86 9C 97 9A 8C 8C 93 ; 瘡摻媽瓠瀾漣殞憲
00002380h: 9A 8B 8C 8F 93 9E 86 9C 97 9A 8C 8C 93 9A 8B 8C ; 臻審撻吮櫛瘡摻媽
00002390h: 8F 93 9E 86 9C 97 9A 8C 8C 93 9A 8B 8C 8F 93 9E ; 瓠瀾漣殞憲臻審撻
000023a0h: 86 9C 97 9A 8C 8C 93 9A 8B 8C 8F 93 9E 86 9C 97 ; 吮櫛瘡摻媽瓠瀾漣
000023b0h: 9A 8C 8C 93 9A 8B 8C 8F 93 9E 86 9C 97 9A 8C 8C ; 殞憲臻審撻吮櫛瘡
000023c0h: 93 9A 8B 8C 8F 93 9E 86 9C 97 9A 8C 8C 93 9A 8B ; 摻媽瓠瀾漣殞憲臻
000023d0h: 8C 8F 93 9E 86 9C 97 9A 8C 8C 93 9A 8B 8C 8F 93 ; 審撻吮櫛瘡摻媽瓠
000023e0h: 9E 86 9C 97 9A 8C 8C 93 9A 8B 8C 8F 93 9E 86 9C ; 瀾漣殞憲摻媽瓠瀾
000023f0h: 97 9A 8C 8C 93 9A 8B 8C 8F 93 9E 86 9C 97 9A 8C ; 櫛瘡摻媽瓠瀾漣殞
```

<http://blog.csdn.net/whklhhhh>

我选中的部分即为循环节，将它们dump下来取反再与0异或即可得到密钥的循环：

```
s = [0x9A, 0x8C, 0x8C, 0x93, 0x9A, 0x8B, 0x8C, 0x8F, 0x93, 0x9E, 0x86, 0x9C, 0x97, 0x9A, 0x8C, 0x8C, 0x
for i in s:
    print(chr((~i ^ 0)%256), end='')
```

```
essletsplaychessletsplaychessletsplaychessletsplaychessletsplaychessletsplaychess
```

可以从中看到”letsplaychess”这个有意义的英文句子

如果密钥无意义的话就只能循环密钥来爆破了

接下来直接通过EXE解密也行，自己写脚本也可以了：

```
key = "letsplaychess"
f = open("ransomware/file", "rb")
data = f.read()
exe = []
for i in range(len(data)):
    exe.append(((~data[i] ^ ord(key[i%len(key)])) % 256))
f.close()

f = open("ransomware/file_de.exe", "wb")
f.write(bytes(exe))
f.close()
```

运行生成的程序就能得到flag了

另外在去花指令的环节，参考得知
还可以用IDAPython脚本
或者直接Python读二进制来去除

```
data = open('run.exe','rb').read()
data = data.replace('\x60\x61\x90\x50\x58\x53\x5b', '\x90\x90\x90\x90\x90\x90\x90')
open('run_dejunk.exe','wb').write(data)
```

(出自<http://www.tuicool.com/articles/fqAZfe6>)

C. 明日计划

Reversing.kr